DEVELOPMENT AND IMPLEMENTATION OF AN EXPERT SYSTEM FOR REMOTELY ACCESSING A RELATIONAL DATABASE

BY

AGUSTIN ORTIZ, JR.

A THESIS SUBMITTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE

UNIVERSITY OF FLORIDA

1988

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188
Exp. Date: Jun 30, 1986

1a. REPORT SECURITY CLASSIFICATION

1b. RESTRICTIVE MARKINGS

2a. SECURITY CLASSIFICATION AUTHORITY

3. DISTRIBUTION/AVAILABILITY OF REPORT

DISTRIBUTION STATEMENT A
Approved for public release;
Distribution Unlimited

2b. DECLASSIFICATION / DOWNGRADING SCHEDULE

4. PERFORMING ORGANIZATION REPORT NUMBER(S)

Development and Implementation of an Expert System to Remotely Access a Relational Database

5. MONITORING ORGANIZATION REPORT NUMBER(S)

Final Report, 21 July 1988

6a. NAME OF PERFORMING ORGANIZATION

6b. OFFICE SYMBOL (*If applicable*)

7a. NAME OF MONITORING ORGANIZATION

HQDA, MILPERCEN

6c. ADDRESS (*City, State, and ZIP Code*)

7b. ADDRESS (*City, State, and ZIP Code*)

Alexandria, VA 22332

8a. NAME OF FUNDING / SPONSORING ORGANIZATION

8b. OFFICE SYMBOL (*If applicable*)

9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER

Student, HQDA, MILPERCEN (DAPC-OPA-E), 200 Stovall St., Alexandria, VA 22332

8c. ADDRESS (*City, State, and ZIP Code*)

10. SOURCE OF FUNDING NUMBERS

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT ACCESSION NO.

11. TITLE (*Include Security Classification*)

HQDA, MILPERCEN, ATTN: DAPC-OPA-E, 200 Stovall St., Alexandria, Virginia 22332

12. PERSONAL AUTHOR(S)

Ortiz, Agustin 21 Jul 88

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (*Year, Month, Day*)	15. PAGE COUNT
Final	FROM _____ TO _____	1988, July, 21	89

16. SUPPLEMENTARY NOTATION

Approved for public release; distribution is unlimited.

17. COSATI CODES		
FIELD	GROUP	SUB-GROUP

18. SUBJECT TERMS (*Continue on reverse if necessary and identify by block number*)

Masters Thesis, University of Florida, 1988

19. ABSTRACT (*Continue on reverse if necessary and identify by block number*)

A knowledge based expert system has been designed to allow a relational database to be queried by a user who has no training on database management systems. A relational database was chosen as an efficient method of storing information about Cub Scout Packs, the Dens that comprise them, the people involved in the administration of each Pack as well as the members of their Dens, their leaders and parents. The database can provide administrative information about the packs or dens, either collectively or individually as well as information about the pack staff or den members. Since more than one member of the same family is usually involved in the same pack, the information common to all members of the family was stored in a table separate from the information that was unique to the individual people. The fact that a person could hold more than one position in the pack made it necessary to create tables to establish the relationships between the pack and its staff and between the den and its members. In order to access the information stored in the database without extensive training about relational databases or structured query languages (SQL), a knowledge-based expert system was designed on the Texas

20. DISTRIBUTION / AVAILABILITY OF ABSTRACT

☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS

21. ABSTRACT SECURITY CLASSIFICATION

UNCLAS

22a. NAME OF RESPONSIBLE INDIVIDUAL

MAJ AGUSTIN ORTIZ

22b. TELEPHONE (*Include Area Code*)

(602) 538-8904

22c. OFFICE SYMBOL

ASQB-SEP-C

DD FORM 1473, 84 MAR

83 APR edition may be used until exhausted.
All other editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE

LTC

(410) 293-2103 JSC-AR

DEVELOPMENT AND IMPLEMENTATION OF AN EXPERT SYSTEM FOR REMOTELY ACCESSING A RELATIONAL DATABASE

Agustin Ortiz, Jr.

Development and Implementation of an Expert System to Remotely Access a Relational Database

MAJ AGUSTIN ORTIZ
HQDA, MILPERCEN (DAPC-OPA-E)
200 Stovall Street
Alexandria, VA 22332

Final Report, 21 July 1988

A thesis submitted to the University of Florida in partial fulfillment of the requirements for the degree of Master of Science.

ACKNOWLEDGMENTS

The author would like to express his appreciation and esteem to his advisor, Dr. A. Antonio Arroyo, for his guidance and support throughout the course of this research, and throughout the majority of the author's graduate studies.

The author is grateful to Dr. Donald G. Childers and Dr. Jose C. Principe for serving on the supervisory committee.

The author is grateful to Captain Richard Routh, Director of the United States Army Artificial Intelligence Training Facility, Fort Gordon, Georgia, for suggesting the project and providing assistance in the development of the expert system.

The author is grateful to Mr. Craig Lanning of the United States Army Artificial Intelligence Training Facility, Fort Gordon, Georgia, for assisting in the implementation of the expert system.

The author is extremely grateful for the love and support of his wife and family throughout his graduate studies and the preparation of this document.

The author is grateful for the technical assistance of the following individuals: Richard McCurdy, Electrical

Engineering Department, University of Florida; Andy Wilcox and Eric Johnson, Computer and Information Sciences Department, University of Florida; Quinton May, Texas Instruments, Incorporated, Tampa, Florida.

The author is extremely grateful to the Almighty God who provided for all of his needs throughout his graduate studies and the preparation of this document.

This research, as well as all of the author's graduate studies, was funded by the Department of the Army under provisions of the fully funded program in accordance with Army Regulation 621-1.

TABLE OF CONTENTS

Page

ACKNOWLEDGMENTS.....ii

LIST OF FIGURES.....v

ABSTRACT.....vi

CHAPTERS

I INTRODUCTION.....1

II DESIGN OF THE DATABASE.....6

III EXPERT SYSTEM DESIGN.....16

Overview of Expert Systems.....16
Automated Reasoning Tool (ART).....17
Scout-Data.....18

IV IMPLEMENTATION OF THE DESIGN.....30

V SYSTEM EVALUATION AND RESULTS.....37

VI CONCLUSION.....42

APPENDICES

A RELATIONAL DATABASE SCHEMA.....45

B EXPERT SYSTEM SOURCE CODE.....51

C USER'S MANUAL FOR EXPERT SYSTEM.....61

D DRIBBLE FILE OF SAMPLE SESSION.....64

E NOTES ON NETWORKING AND REMOTE PROCEDURE CALLS..69

REFERENCES.....77

BIOGRAPHICAL SKETCH.....80

LIST OF FIGURES

FIGURES

2-1 The FAMILY File (Table).....7

2-2 Relational Database Schema (Tables).....8

2-3 Database Schema (Types).....9

2-4 Entity-Relationship (ER) Diagram.....14

2-5 Results of Example Query.....15

3-1 Expert System Flowchart.....20

E-1 Computer Communications Architecture.....70

E-2 Network Reference Models and Layering.....72

E-3 DOD Military Standard Protocols.....72

Abstract of Thesis Presented to the Graduate School of the University of Florida in Partial Fulfillment of the Requirements for the Degree of Master of Science

DEVELOPMENT AND IMPLEMENTATION OF AN EXPERT SYSTEM FOR REMOTELY ACCESSING A RELATIONAL DATABASE

BY

AGUSTIN ORTIZ, JR.

August 1988

Chairman: A. Antonio Arroyo
Major Department: Electrical Engineering

A knowledge based expert system has been designed to allow a relational database to be queried by a user who has no training on database management systems. A relational database was chosen as an efficient method of storing information about Cub Scout Packs, the Dens that comprise them, the people involved in the administration of each Pack as well as the members of the Dens, their leaders and parents. The database can provide administrative information about the packs or dens, either collectively or individually as well as information about the pack staff or den members. Since more than one member of the same family is usually involved in the same pack, the information common to all members of the family was stored in a table separate from the information that was unique to the individual people. The fact that a person could hold more than one position in the pack made it

necessary to create tables to establish the relationships between the pack and its staff and between the den and its members. In order to allow an individual to access the information stored in the database without extensive training about relational databases or structured query languages (SQL), a knowledge-based expert system was designed on the Texas Instruments Explorer using the Inference Corporation's Automated Reasoning Tool (ART). The system uses rules based on the designer's expert knowledge of the database structure and Structured Query Language (SQL) to determine the exact query that is desired, then prints the required query on the screen and issues the query through an ETHERNET network to the database located on another computer. The major objective of this research is to take advantage of the capabilities of an existing "artificial intelligence" machine to act as an interface between the average human user and the database management system on another computer.

CHAPTER I
INTRODUCTION

This thesis proposes the use of an expert system on a Lisp-based artificial intelligence computer to act as the interface between a human user and a relational database on another computer connected via an Ethernet to the first computer. The purpose of the expert system is to allow a user with no training in database management systems to obtain information from the database.

During the past ten years there has been an increase in the availability of relational database management systems [Da86, Gr87]. This has caused many organizations to store data in relational databases. These same organizations have had to spend time and money training people on the use of the database management systems. It would greatly benefit any organization that used a database management system if there were a way for their employees to use the database without the need for extensive training. During the past ten years there also has been research done in the field of artificial intelligence on the use of rule-based or knowledge-based expert systems to execute tasks previously requiring the expertise of a human specialist [Ch87, Cl87, Ge83, Ha85, Ha83, Le85, Mc82, Pr85, St82, Wa86, Wi84]. Some research was done combining the two

fields of database management and artificial intelligence to produce intelligent databases[Da86, Gr87].

A database management system provides a means of storing and retrieving data in order to obtain information. The data, itself, is stored in files (also referred to as tables)[Gr87]. The user may be allowed to perform the following operations on the database:

Add new files to the database

Insert data into existing files

Retrieve data from existing files

Update the data in an existing file

Delete data from an existing file

Remove existing files from the database

A database can provide an organization centralized control of operational data while allowing many users to access the data simultaneously[Da86]. In order to avoid anomalies caused by functional dependencies between the attributes of a table in the database, it is necessary to normalize the database. A relational database is considered to be in third normal form (3NF) if all of the attributes in one table that could determine an attribute of another table are keys or all of the attributes of the second table are part of the key of some table. This aspect was used to determine which data was to be stored in each table. Access to the information in the database is usually accomplished via a high level Data Manipulation Language (DML). Structured Query Language (SQL) has been

established as the standard language for relational databases by the American National Standards Institute (ANSI)[Gr87].

The principle behind expert systems is the use of a knowledge base to produce intelligent behavior modeling that of the human[Ch87, Ge83, St82]. It is desirable to have the system interact with the user in the same way that another human would[Ch87]. The system used in this project provides a "rule-driven, mixed-initiative inter-face" allowing the user and the system to exchange responses while the program runs. This means that sometimes the computer is answering the user's questions and sometimes the user is answering the computer's questions[Cl87, Vol. 4, 3]. The user is also able to modify the knowledge base at any time while the program is running, if desired[Ch87, Cl87]. The ultimate success of the expert system in accomplishing the desired objective will depend on the ability of the knowledge engineer to extract the information on which to base the rules from the expert on the subject domain[Cl87]. In this project, the author has played the role of subject matter expert as well as knowledge engineer. The rules of the expert system consist of "if conditions then action-list1 else action-list2" structures[Cl87, Wa86, Wi84]. At any time that the conditions specified by a rule are true in the knowledge base, that rule is triggered. Many rules could be triggered by the same set of conditions[Cl87, Wi84]. Some

method must be used to determine which of the triggered rules will actually fire[Wi84]. When a rule fires, it will take the specified actions in accordance with the appropriate action-list. This could alter the facts in the knowledge base, taking some rules off of the triggered list (agenda) and adding others[Cl87, Wi84]. This process will continue as long as the system is active. The command to halt may come from a user or be part of the action-list of one of the rules[Cl87].

In order to be able to access a database on one computer from another computer, it is necessary that the two machines be interconnected via some type of network (hardware) and have a common protocol (software)[Em87, Ra87, Sp87, St87a, St87b, St88]. In this case, the two computers were interconnected to a local area network via an ethernet and were able to communicate using Transmission Control Protocol/ Internet Protocol (TCP/IP) and Sun Microsystems Remote Procedure Call (RPC).

In this project, a relational database was designed to efficiently store data about Cub Scout packs, the dens comprising each pack and the people associated with both. The next chapter discusses the details of the database design. An expert system was designed to determine the appropriate query syntax needed to obtain the information desired by the user. Chapter III discusses the design of the expert system. Chapter IV discusses the implementation of the expert system and Chapter V provides an evaluation

of the system and discusses the results of the research to date. Chapter VI presents conclusions drawn from the current research and provides a projection of further research that is desirable.

CHAPTER II
DESIGN OF THE DATABASE

In recent years, businesses and organizations have realized that information can be very valuable to them[Gr87]. This has resulted in extensive research into the development of database management systems (DBMS) to allow data to be stored conveniently and retrieved to provide the needed information[Gr87, Da86]. As might be expected, these DBMS are complex software packages that usually require extensive user training. There are three major database models in use today: network, hierarchic and relational. They differ in the way relationships between files are represented. Although it is beyond the scope of this paper to explain the first two models, it must be stated here that the network and hierarchic models allow one-to-many relationships to be represented directly in a data structure diagram, while the relational database model represents relationships implicitly in the file attributes[Gr87]. To illustrate the basic concepts of computerized databases, the extract of the FAMILY file in figure 2-1 will be used. This file contains some information about the families associated with a certain Cub Scout pack.

family_number	last_name	street_addr	city	state	zip_code	area_code	number
1	Barnett	1121 NW 34th St	Gville	FL	32605	904	
60	Ortiz	3215 NW 53rd AV	Gville	FL	32605	904	371-1930
2	Belyew	2001 Space Odys	Gville	FL	32606	904	
10	Jones	2835 NW 41st PL	Gville	FL	32605	904	
11	Davis	3444 NW 50th Wa	Gville	FL	32605	904	378-6199
20	Arrants	6102 NW 33rd St	Gville	FL	32605	904	375-3921

Figure 2-1. The FAMILY File (Table)

Files. The file is the basic component of the database system. Files contain data and are also referred to as relations or tables. The row across the top of the table gives information about the types of objects present in the file. The elements of this row are the attributes or field names. The actual data from the database is found in the remaining rows of the table, with the data in each column corresponding to the type of the attribute. Since it is common to visualize files in the two-dimensional manner displayed above, with rows and columns, they are usually referred to as tables[Gr87].

Schema. The schema of a database is a description of the structure of its tables. This description includes the names of the attributes, their data types, and the relationships between tables of the database. This is contrasted with an instance of the database, which is a description of the contents of the files of the database. Each row of a table is called a record. In Figure 2-1, above, the first record is

1|Barnett |1121 NW 34th St|Gville|FL | 32605| 904| .

The individual elements of the record (e.g. 1, Barnett) are called fields or field values. The DBMS allows the user to store data in tables and later access the data from more than one table in order to obtain information. In order to develop the database, a conceptual view or schema must be constructed to define the tables, the attributes of each table, the data types of the attributes and the data integrity constraints. The schema for our database is located in Figures 2-2 and 2-3 below.

Tables	Attributes (Key Fields Underlined)
Pack	NUMBER, CHARTER_ORG, CHARTER_DATE, COUNCIL
Den	DEN_NUMBER, PACK_NUMBER, MEETING_NIGHT, MEETING_LOCATION
People	PERSON_NUMBER, FAMILY_NUMBER, FIRST_NAME, MI, WORK_PHONE
Family	FAMILY_NUMBER, LAST_NAME, STREET_ADDR, CITY, STATE, ZIP_CODE, AREA_CODE, NUMBER
Staff	PERSON_NUMBER, FAMILY_NUMBER, PACK_NUMBER, POSITION
Members	PERSON_NUMBER, FAMILY_NUMBER, DEN_NUMBER, PACK_NUMBER, POSITION

Figure 2-2. Relational Database Schema (Tables)

Types	Attributes
DATE	CHARTER_DATE
NUMERIC (1)	DEN_NUMBER
NUMERIC (3)	PACK_NUMBER, FAMILY_NUMBER, AREA_CODE
NUMERIC (5)	ZIP_CODE
NUMERIC (9)	PERSON_NUMBER
STRING (1)	MI
STRING (2)	STATE
STRING (3)	MEETING_NIGHT
STRING (5)	POSITION
STRING (8)	NUMBER, WORK_PHONE
STRING (10)	CITY
STRING (11)	LAST_NAME, FIRST_NAME, MEETING_LOCATION
STRING (15)	COUNCIL, STREET_ADDR
STRING (22)	CHARTER_ORG

Figure 2-3. Database Schema (Types)

There are three types of relationships possible between two files of a database: one-to-one, one-to-many, and many-to-many. In a one-to-one relationship, there is only one record in a file corresponding to each record in the other file where the two records have a common value in one of the fields. There are no examples of this relationship in our database. In a one-to-many relationship, many records from the second file may be associated with each record of the first file. The relationship between the pack table

and the den table is such a relationship -- there may be many dens in each pack. In the many-to-many relationship, there may be many records from the second file associated with each record of the first file and many records from the first file may be associated with each record of the second file[Gr87]. In our database, there are many people associated with each den and some people are associated with more than one den.

The Cub Scout Database. The database used for this project was kept relatively simple in order to focus on the aspects of expert system design and networking. The rest of this chapter explains the basis for the database design. The Cub Scout pack is an organization found in most communities which has parallels in its structure to larger organizations.

Integrity. Integrity is a measure of the correctness of the data in the database at any given time. The primary integrity constraint used is called a key constraint[Gr87]. It is used to insure that two records are not allowed to exist in a table with the same value in the key field. For instance, there should not be two dens with the same number in the same pack. The other integrity constraint that is sometimes allowed by a DBMS is that a field may not be NULL, in other words, it must have a value. The Unify Corporation's Unify relational database management system (Unify) enforces this constraint for key fields only[Un85a, Un85b, Un85c].

Relational Database. In a relational database, a table is considered to be a relation, where a relation is a subset of a set of elements. Operations are performed on the relations based on the relational algebra to obtain data from related tables. The relational databases are easier to use than the other two types because of the ability to perform these operations[Gr87]. The most significant operations are listed below:

Projection	omit some of the columns
Selection	omit some of the rows
(Natural) Join	obtain the product of two tables

Relations. Each pack has a number to distinguish it from other packs in the same council. This number was used as the key for the pack relation. Each pack is chartered to an organization in the local community. Thus, the attributes associated with the pack, itself, are its number, charter organization, the date it was chartered and the council it is associated with. The actual active units in which the Cub Scouts participate are the dens. There are usually several dens in a pack. Each den has a number unique within the pack. Other information that might be stored about the den includes the night and location that meetings are held. By making the pack number one of the attributes of the DEN table, the dependency of the den on the pack is established. The pack is a family-oriented organization[Bo86]. It is administered by a committee that usually is composed of parents of scouts. The den leader may also be the parent of one of the boys in the den.

Additionally, there may be boys from the same family in different dens. In order to avoid duplicating the information that is common to the family (e.g. last name, address and home phone number) for each member of the family, two tables were created. The FAMILY table contains the information common to the entire family and is distinguished by a unique three-digit family number. The PEOPLE table contains only the information that is unique to the individual person (first name, middle initial and work phone number) in addition to the family number and a unique person number for easy retrieval of information about the individual. The family number establishes the dependency of the individual person on the family.

Relationships. The FAMILY table and the PACK table are independent of each other so there is no implicit relationship. In order to establish the relationships between the people in the database and the appropriate organization, tables had to be created that made the relationship explicit. The MEMBERS relationship allows the members of a den to be identified by their person number, family number, their position in the den (e.g. scout, leader, parent), the den number and pack number. The STAFF relationship identifies pack committee members similarly by their person number, family number, committee position (e.g. cubmaster, committee chief, etc.) and the pack number. By storing this information in separate tables, a person's position can be changed without interfering with

the personal or family information. The relationships between tables are often depicted in an Entity-Relationship (ER) diagram like the one in Figure 2-4 below. In the ER diagram, the rectangular boxes represent the tables or relations (entities), the diamonds represent relationships, and ovals represent attributes[Gr87, Da86]. Straight lines are used to represent connections between tables, with 1 and N written on the line to represent a one-to-many relationship between connected files. The key attributes are usually underlined and weak entities (those that only have meaning with respect to an instance of the connected table) are in double boxes[Gr87]. In our database, people have no meaning without a corresponding family and dens cannot exist without a corresponding pack.

Queries. By joining tables on shared attributes, it is possible to obtain information about one or more packs, to include any or all of the following: the first and last names, middle initial, home address, home and work phone numbers of committee members. Similarly, any information in the database may be obtained about a den member in addition to the administrative data about the den, itself. For example, to obtain the first names, last names, addresses and home phone numbers of all the members of den 3, pack 83, the following SQL query would be used:

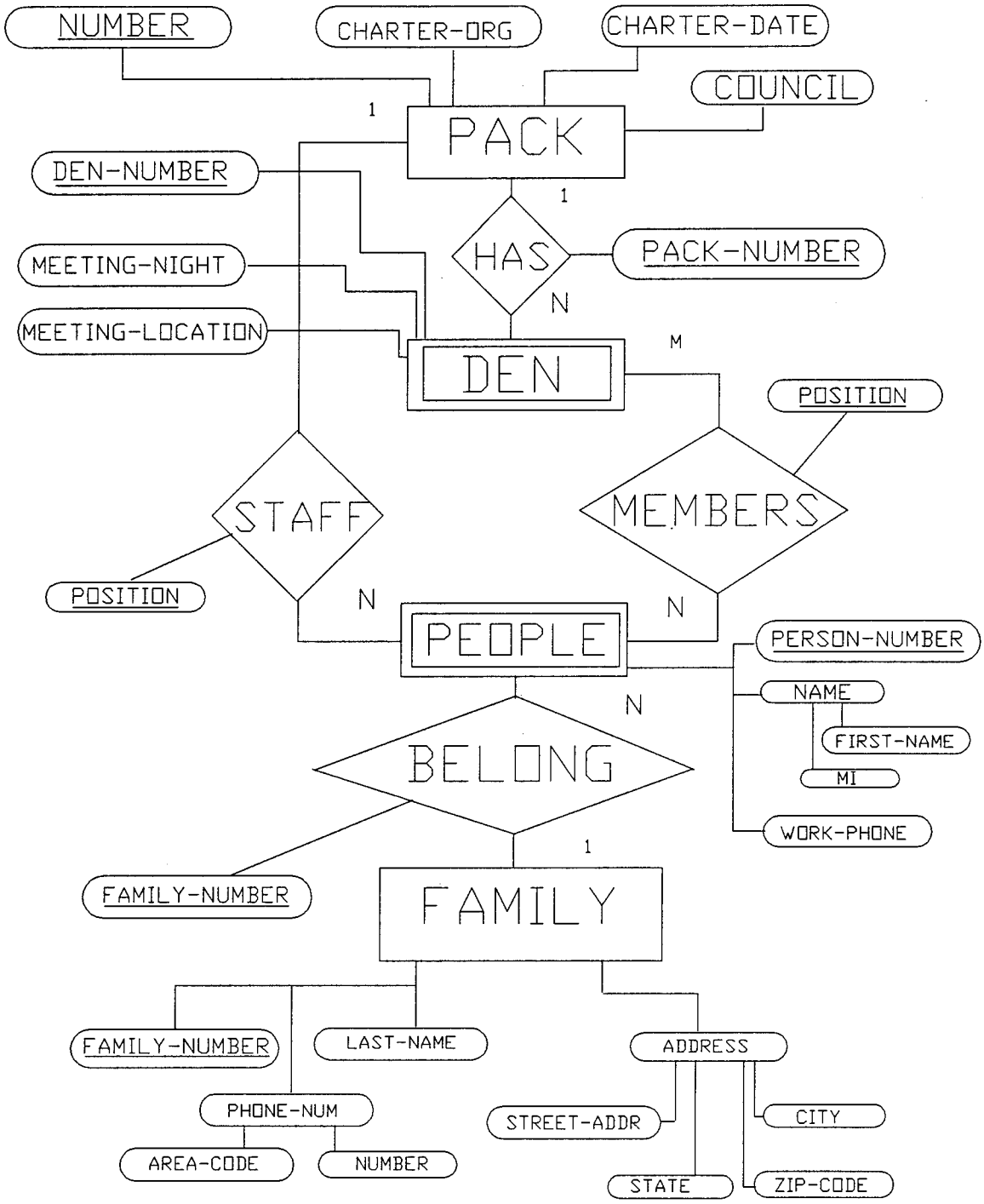

Figure 2-4. Entity-Relationship (ER) Diagram

```
select unique first_name, last_name, street_addr, city, number
from members m, people p, family f
where m.person_number=p.person_number
and m.family_number=p.family_number
and m.family_number=f.family_number
     and m.pack_number=83
     and m.den_number=3
     order by last_name, first_name/
```

The results of the example query are shown in Figure 2-5, below.

```
first_name |last_name  |street_addr    |city      |number
-----------------------------------------------------------

Jason      |Davis      |3444 NW 50th Wa|Gville    |378-6199
John       |Davis      |3444 NW 50th Wa|Gville    |378-6199
Linda      |Davis      |3444 NW 50th Wa|Gville    |378-6199
Agustin    |Ortiz      |3215 NW 53rd AV|Gville    |371-1930
Carlos     |Ortiz      |3215 NW 53rd AV|Gville    |371-1930
Kathy      |Ortiz      |3215 NW 53rd AV|Gville    |371-1930
```

Figure 2-5. Results of Example Query

The complexity of the above query illustrates the need for extensive training to enable a user to execute queries involving multiple tables. The user would need to know in advance the attributes of each table and which ones were keys. Additionally, the user must know the syntax of SQL.

CHAPTER III
EXPERT SYSTEM DESIGN

This chapter contains background information about the history and characteristics of expert systems. Rule-based systems are the basis of the most popular paradigm used for solving problems in knowledge engineering. It is this branch of Artificial Intelligence that specializes in in building expert systems. There are many examples of rule-based systems that have proven themselves. Among them are XCON, MYCIN, and PROSPECTOR[Wi84].

Overview of Expert Systems

Knowledge-based expert systems have been in use since the mid-1960s. They have been used for the performance of tasks that normally require a professional specialist. The expert system, itself, is a special-purpose computer program constructed to handle problems within a narrow domain[Ch87]. It depends on the expertise of one or more human experts as extracted by a knowledge engineer. This expert advice is coded as a series of rules and applied to a special knowledge base that contains information about a situation. The situation may be either real or hypothetical. The action that a human expert would take given the same circumstances are simulated by the actions specified by the rules of the system. Although the

computerized expert system is limited in that it cannot be programmed to do something that no human knows how to do, it has many advantages over the human expert. The properly programmed expert system can use the combined knowledge of several experts. It is fully informed at all times, considering all facts in the knowledge base before taking any action[Cl87]. Expert systems are not subject to human emotions, physical fatigue, or illness. Since the actions taken depend entirely on the facts (knowledge), it is often said that the real power of the expert system is a function of the knowledge it contains[Ch87, Cl87, Wi84]. The major issue in the development of an expert system is the construction and manipulation of the knowledge base. Expert systems usually consist of the following components: a knowledge base, a data base, a control mechanism, and a knowledge-base editor. The knowledge can be separated into modules and the control mechanism tries to match data from the data base to the knowledge base[Ch87]. The knowledge editor is a user interface that allows the user to modify the data in the knowledge base.

Automated Reasoning Tool (ART)

Inference Corporation's ART can be used to develop expert systems. ART provides the knowledge engineer with the following capabilities:

Rule-based programming
Forward chaining
Backward chaining
Schema-based knowledge representation, allowing a program to reason about the relationships between objects
A method of modeling situations that change dynamically using viewpoints

An interactive environment for developing and debugging the system.
A graphics interface package[Cl87].

Scout-Data

Overview. The Scout-Data expert system uses a series of facts and rules to determine the proper SQL query needed to obtain the desired information. For the purpose of this project, the number of queries was limited to some commonly useful ones. The system is initialized in an unknown state. It must ask the user if he wants to query the database in order to determine whether or not it is finished. If the user wants to query the database, the system asks if the desired information is about packs or dens. These are the two types of information of interest to a user of the database. With respect to the pack, there are two types of information of interest: administrative and staff. Administrative information is found in the pack table, itself, while staff information identifies the committee members (people) associated with the pack. With respect to the dens, there are likewise two types of information: administrative information contained in the den table and information identifying the members of the den. The expert system must determine the type of query to be executed by asking the user which type of information is desired. Once the type of query has been determined, it is possible to get the information about all of the units in the database or only some selected units. Again, the system asks the user a series of questions in order to determine which query to execute. After the query has been

executed, the system will ask the user whether or not another query is desired and react accordingly. Figure 3-1 below is a flowchart depicting the flow of control in the system.

Facts. The initial flag asserts dummy facts to insure that the appropriate rules are triggered to obtain information from the user. The required facts are as follows:

use-db	whether or not to query the database (y/n)
type-query	information about pack (p) or den (d)
type-info	administrative info (a), info about pack staff (s) or den members (m)

```
(deffacts initial-flag              ;Initial facts used to
                                    ;start the system
  (use-db unknown)
  (type-query  unknown)
  (type-info unknown)
)
```

Schemata. The term schema, when used in relation to ART, has a meaning different from that which has been used above in discussing the relational database. An ART schema is a collection of facts about a particular object that has been given a name. This will allow ART to reason about the facts that relate to the object. The definition parallels that of a semantic net. The definition of schemata provides symbols to represent objects within the knowledge base. Inheritance relations are used to bind schemata into semantic nets[C187]. The following schemata were defined to establish the semantic network representing

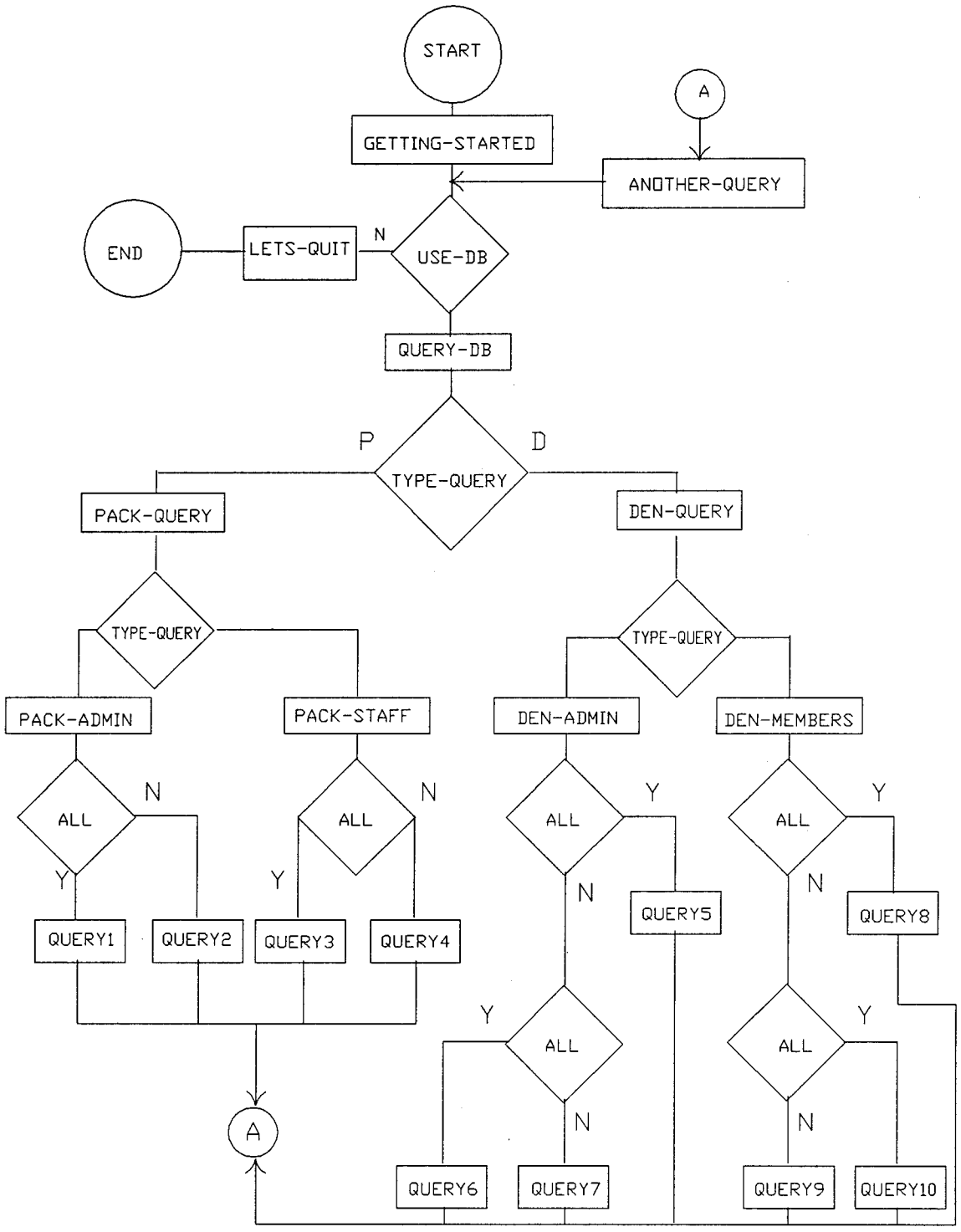

Figure 3-1. Expert System Flowchart

the relationships between the tables of the scout database.

Subclass-of. Subclass-of is used to establish the inheritance of attributes from the class to its subclasses. In this system, this allows the den schema to inherit the information from the pack schema and the people schema to inherit information from the family schema.

```
(defschema subclass-of
      (instance-of inh-relation)
      (new-relations (is-a (?domain)(subclass)))
      (inverse has-subclasses))
```

Has-subclasses. This is the inverse of subclass-of.

Pack. Pack has slots for each of the attributes of the pack relation.

Den. Den has slots for each of the attributes of the den relation.

People. People has slots for each of the attributes of the people relation.

Family. Family has slots for each of the attributes of the family relation.

Rules. Described below are the rules used by the expert system. The commented source code containing all of the rules is located in Appendix B.

Getting-started. This rule retracts the dummy use-db fact, displays a greeting and asks the user if the database is to be queried. Asserts a new use-db fact based on the user response.

```
(defrule getting-started
   ?x<-(use-db ~Y & ~N)        ;WHETHER to query the DB or
                               ;NOT unclear or unknown.
```

```
;(Should only occur at the beginning of the session.)
=>
   (set-interactive-mode nil)    ;turn off warning messages
   (retract ?x)                  ;remove unsatisfactory fact.
   (select-window 'command-window)
   (reshape-window #L'command-window 10 10 660 300)
   (clear-window #L'command-window)
   (printout t t t                          ;output to screen.
        "Hello.  My name is ART.  I can help you access the
          Cub Scout Data Base." t t)
   (if (y-or-n-p (format nil "Do you wish to query the Data
Base? "))
       then
       (assert (use-db Y))
       else
       (assert (use-db N))))
```

Query-db. In the presence of a (use-db Y) fact and a dummy type-query fact, this rule retracts the type-query fact, then asks the user whether information is desired about packs or dens. Asserts a new type-query fact based on the user response.

```
(defrule query-db
      "An SQL query is desired"
  (use-db Y)
  ?x<-(type-query ~P&~D)
=>
(retract ?x)
(printout t t t "Do You want information about a pack [P]
or a den [D]?")
(assert (type-query =(read))))
```

Pack-query. In the presence of a (use-db Y) fact, a (type-query P) fact and a dummy type-info fact, this rule retracts the dummy fact, then asks the user whether information is desired about the pack or its staff. Asserts a new type-info fact based on the user response.

```
(defrule pack-query
    "determine whether info is desired about the pack,
itself, (admin)
    or about the people that run it (staff)"

  (use-db y)
  (type-query P)
```

```
  ?x<-(type-info ~A&~S)
=>
(retract ?x)
(printout t t "Do you want administrative information about
the pack [A]
or information about the pack staff [S]? ")
(assert (type-info =(read))))
```

Den-query. In the presence of a (use-db Y) fact, a (type-query D) fact and a dummy type-info fact, this rule retracts the dummy fact, then asks the user whether information is desired about the den or its members. Asserts a new type-info fact based on the user response.

```
(defrule den-query
    "determine whether info is desired about the den,
itself, (admin)
    or about the people associated with it (members)"

  (use-db y)
  (type-query D)
  ?x<-(type-info ~A&~M)
=>
(retract ?x)
(printout t t "Do you want administrative information about
the den [A]
or information about the den members [M]? ")
(assert (type-info =(read))))
```

Pack-admin. In the presence of a (use-db Y) fact, a (type-query P) fact and a (type-info A) fact, this rule asks the user whether the information is desired about all packs in the database. If the user response is Y, the specified query is displayed on the screen with its output below it. Otherwise, the user is asked for the number of the pack in order to generate the query as described above. Callrpc is used to remotely execute the query on the host computer.

```
(defrule pack-admin
    "User wants admin info about the pack"
(use-db Y)
```

```
(type-query P)
(type-info A)
=>
(reshape-window #L'command-window 10 10 660 300)
(if (y-or-n-p (format nil "Do you want the information on
all packs? "))
    then
    (clear-window #L'command-window)
    (printout t t "The required SQL query is"
         t t "select *"
         t t "from pack"
         t t "order by number/" t t)
    (printout t t "Here is the information you requested:"
t t)
    (reshape-window #L'command-window 10 10 660 450)
;*********************
    (callrpc "beach" 20118651 1 1 :xdr_void nil :xdr_string
"lm:gus;packs.text" :udp)
;*********************
    (view-file "lm:gus;packs.text")
    else
    (printout t t "Please enter the Pack number [0-999]: ")
     (bind ?number (read))
     (clear-window #L'command-window)
     (printout t t "The required SQL query is"
         t t "select *"
         t t "from pack"
         t t "where number=" ?number"/" t t)))
```

Pack-staff. In the presence of a (use-db Y) fact, a (type-query P) fact and a (type-info S) fact, this rule asks the user whether the information is desired about all packs in the database. If the user response is Y, the specified query is displayed on the screen with its output below it. otherwise, the user is asked for the number of the pack in order to generate the query as described above.

```
(defrule pack-staff
    "User wants staff info about the pack"
(use-db Y)
(type-query P)
(type-info S)
=>
(reshape-window #L'command-window 10 10 660 300)
(if (y-or-n-p (format nil "Do you want the information on
all packs? "))
    then
    (clear-window #L'command-window)
```

```
    (printout t t "The required SQL query is"
         t t "select pack_number, position, first_name, MI,
last_name, number, work_phone"
         t t "from staff s,people p, family f"
         t t "where s.person_number=p.person_number and"
         t t "s.family_number=p.family_number and"
         t t "s.family_number=f.family_number"
         t t "order by pack_number/" t t)
    else
    (printout t t "Please enter the Pack number [0-999]: ")
     (bind ?pnum (read))
     (clear-window #L'command-window)
     (printout t t "The required SQL query is"
         t t "select pack_number, position, first_name, MI,
last_name, number, work_phone"
         t t "from staff s,people p, family f"
         t t "where s.person_number=p.person_number and"
         t t "s.family_number=p.family_number and"
         t t "s.family_number=f.family_number and"
         t t "pack_number=" ?pnum"/" t t)))
```

Den-admin. In the presence of a (use-db Y) fact, a (type-query D) fact and a (type-info A) fact, this rule asks the user whether the information is desired about all dens in the database. If the user response is Y, the specified query is displayed on the screen with its output below it. otherwise, the user is asked for the number of the pack. Then asks the user whether the information is desired about all dens in the pack. If the user response is Y, the specified query is displayed on the screen with its output below it. Otherwise, the user is asked for the number of the den in order to generate the query as described above.

```
(defrule den-admin
    "User wants admin info about the den"
(use-db Y)
(type-query D)
(type-info A)
=>
(reshape-window #L'command-window 10 10 660 300)
(if (y-or-n-p (format nil "Do you want the information on
all dens in the DB? "))
```

```
    then
    (clear-window #L'command-window)
    (printout t t "The required SQL query is"
         t t "select *"
         t t "from den"
         t t "order by pack_number,den_number/" t t)
    else
    (printout t t "Please enter the Pack number [0-999]: ")
     (bind ?pnum (read))
     (if (y-or-n-p (format nil "Do you want the information
on all dens in the pack? "))
    then
    (clear-window #L'command-window)
    (printout t t "The required SQL query is"
         t t "select *"
         t t "from den"
         t t "where pack_number=" ?pnum
         t t "order by den_number/" t t)
    else
    (printout t t "Please enter the Den number [0-9]: ")
    (bind ?dnum (read))
    (clear-window #L'command-window)
    (printout t t "The required SQL query is"
         t t "select *"
         t t "from den"
         t t "where pack_number=" ?pnum " and"
         t t "den_number=" ?dnum"/" t t))))
```

Den-member. In the presence of a (use-db Y) fact, a (type-query D) fact and a (type-info M) fact, this rule asks the user whether the information is desired about all dens in the database. If the user response is Y, the specified query is displayed on the screen with its output below it. otherwise, the user is asked for the number of the pack. Then asks the user whether the information is desired about all dens in the pack. If the user response is Y, the specified query is displayed on the screen with its output below it. Otherwise, the user is asked for the number of the den in order to generate the query as described above.

```
(defrule den-member
    "User wants member info about the den"
```

```
(use-db Y)
(type-query D)
(type-info M)
=>
(reshape-window #L'command-window 10 16 780 350)
(if (y-or-n-p (format nil "Do you want the information on
all dens? "))
    then

    (clear-window #L'command-window)
    (printout t t "The required SQL query is"
         t t "select den_number, pack_number, position,
first_name, MI, last_name, number, work_phone"
         t t "from member m,people p, family f"
         t t "where m.person_number=p.person_number and"
         t t "m.family_number=p.family_number and"
         t t "m.family_number=f.family_number"
         t t "order by
pack_number,den_number,position,last_name/" t t)
    else
    (printout t t "Please enter the Pack number [0-999]: ")
     (bind ?pnum (read))
     (if (y-or-n-p (format nil "Do you want the information
on all dens in the pack? "))
    then

    (clear-window #L'command-window)
    (printout t t "The required SQL query is"
         t t "select den_number, pack_number, position,
first_name, MI, last_name, number, work_phone"
         t t "from member m,people p, family f"
         t t "where m.person_number=p.person_number and"
         t t "m.family_number=p.family_number and"
         t t "m.family_number=f.family_number and"
         t t "pack_number=" ?pnum
         t t "order by den_number,position,last_name/" t t)
    else
    (printout t t "Please enter the Den number [0-9]: ")
    (bind ?dnum (read))

    (clear-window #L'command-window)
    (printout t t "The required SQL query is"
         t t "select den_number, pack_number, position,
first_name, MI, last_name, number, work_phone"
         t t "from member m,people p, family f"
         t t "where m.person_number=p.person_number and"
         t t "m.family_number=p.family_number and"
         t t "m.family_number=f.family_number and"
         t t "pack_number=" ?pnum " and"
         t t "den_number=" ?dnum
         t t "order by position,last_name/" t t))))
```

Another-query. In the presence of a (use-db Y) fact, a type-query fact and a type-info fact, this rule asks the user whether or not another query is desired. If the user response is Y, all of the facts are retracted, (use-db Y) is asserted, together with two dummy facts. Otherwise, the use-db fact is retracted and (use-db N) is asserted. This rule will only fire after all other rules with a higher salience (priority) have fired, because it has been assigned a salience of -1.

```
(defrule another-query
    "Find out if user wants to make another query."

  (declare (salience -1))      ;make this wait until other
rules have fired
  ?use-db<-(use-db Y)
  ?type-query<-(type-query ?D & ~unknown)
  ?type-info<-(type-info ?M & ~unknown)
=>

(if (y-or-n-p (format nil "Do you wish to query the Data
Base again? "))
       then
       (retract ?use-db
      ?type-query
      ?type-info)
       (clear-window #L'command-window)
       (assert (use-db Y)
          (type-query  unknown)
          (type-info unknown))
       else
       (retract ?use-db)
       (assert (use-db N))))
```

Lets-quit. In the presence of a (use-db N) fact, this rule halts execution and resets the knowledge base.

```
(defrule lets-quit
    "User doesn't want to query the data base"
(use-db N)
=>
(reshape-window #L'command-window 19 16 398 166)
(clear-window #L'command-window)
(printout t t "Goodbye, have a nice day.")
```

```
(halt)
(reset))
```

CHAPTER IV
IMPLEMENTATION OF THE DESIGN

Database. The Cub Scout Database described in Chapter II and Appendix A was implemented on the Computer and Information Sciences (CIS) department's Gould Powernode using the Unify Corporation's Unify relational database management system. The database resides in the /cisg/grad/gus/scouts subdirectory. In order to access the database directly, the user must either set the DBPATH environment variable to this path or change directory to this subdirectory before entering the "unify" command from the operating system shell. Alternately, SQL queries and updates may be executed from the shell by entering "SQL" followed by the name of a file containing a valid query. If there is no query file or the user wishes to type the query interactively, "SQL" may be entered without arguments. The "sql=>" prompt will be displayed to indicate that the system is ready for a query. Queries must end with a slash(/). After the queries have been completed, the "end" command will return the user to the operating system shell. Unify also provides a Host Language Interface which allows a programmer to write a program in the C language to access the database[Un85a].

Expert System. The expert system described in Chapter III was implemented on the Applied Artificial Intelligence Laboratory (AAIL) Texas Instruments Explorer using the source code contained in Appendix B. The system was compiled using Inference Corporation's Automated Reasoning Tool (ART), version 3.0. A user's manual is provided at Appendix C.

Communications Interface. The objective of this project requires that the expert system be able to communicate with the database. This is possible because both machines are connected to a broadband coaxial cable Radio Frequency (RF) Ethernet. The University of Florida campus has a computer network known as UFNET that provides several network services on a bidirectional RF cable. Several of the computers used by the Electrical Engineering (EE) department on the fourth floor of the Computer Science and Engineering building (CSE) are interconnected via an RF Ethernet operating according to published Institute of Electrical and Electronics Engineers (IEEE) protocol. The EE segment was connected to other segments by a bridge device[Mc88]. It is this bridge that allows the AAIL to communicate with the CIS Gould computer. In order for the expert system to execute an SQL query against the Unify database on behalf of a user, the system must access the transport layer of the Transmission Control Protocol (TCP) or use the Sun Microsystems Remote Procedure Call (RPC) to call a remote procedure server on the remote host[Su86,

St88, Te87]. The transport layer is one of the seven layers of the International Standards Organization (ISO) Open Systems Interconnection (OSI) Model. Its purpose is to provide for the exchange of data between processes in different systems by ensuring that data units are delivered error-free, in sequence and without loss or duplication[St88]. In order to access the transport layer directly from the expert system, a special protocol must be written. This protocol would be similar to the TELNET protocol currently used for remote terminal operations, but would take its input from a file of command specifications instead of from the user's keyboard[St88, Te87].

Remote Procedure Call. The remote procedure call model uses two processes to accomplish a procedure call--the caller's process and the server process on the remote host. The calling process must send a message containing the procedure's identification parameters to the server and wait for a response containing the result of the procedure. The calling process resumes execution upon receipt of the server's message. Although the RPC protocol is independent of the transport protocol, the reliability will be improved by using TCP/IP. RPC provides the means for the user to authenticate into the remote system as well as the means of uniquely identifying the remote procedure to be called[Su86]. The call message must specify the remote program number, version number, and procedure number as well as the external data representation of the input

and output data streams. RPC is not a commonly used procedure and there is not much local expertise about how to implement the actual call. For this reason, this aspect of the project required extensive research and assistance from agencies outside the university. The need to interface two totally different computers using different operating systems (UNIX versus Lisp with FLAVORS) introduced many interesting issues, some of which are still being researched. The fact that Unify is a proprietary product and its source code is not available to the author caused a problem with the use of the RPC. The Host Language Interface made it possible to write a program to access the database. The individual functions of this C program can be registered with RPC for use by the remote machine. The expert system's rules must use the callrpc function to call the registered function. The requirements of network communications and RPC are discussed further in Appendix E. The following code segment was used to test the use of the Unify Host Language Interface with registerrpc:

```
/* * * * * *
sample.c

UNIFY functions Used:
     bseqacc()
     bgfield()
     iniubuf()
     bfaccess()

Based on the bseqsimpl.c example in the Unify Programmers' Manual.

This C program uses the Unify Host Language Interface and the Sun Microsystems
Remote Procedure Call (RPC) to enable a remote user to access all records of the
pack table from the Cub Scout database.     */
```

```
/* include Database header file */

$include "/cisg/grad/gus/scouts/file.h"

#include <stdio.h>
#include "/sys/rpc/rpc.h"
#include "/sys/rpc/xdr.h"

#define SIZE 8192     /* the size of the buffer */
#define program_number          0x20118651
#define version_number          1
#define procedure_number        1

int  *current_query();

rpc_initialize()

{    /* the following procedure call is used to register the program and
procedure with RPC so that it may be called from a remote machine.*/

     if (registerrpc (program_number,
                      version_number,
                      procedure_number,
                      current_query,
                      xdr_void,
                      xdr_int) != 0) {
     printf ("Error in registerrpc\n");
     }
}

int *current_query()
{
     int current_query;

     char   *malloc(),
            *buffer,
            print_buffer[80];

     int    line_number;

/*   Allocate SIZE bytes of memory, where buffer is a pointer
     to the designated memory location (buffer).*/

     if ((buffer = malloc(SIZE)) == (char *) 0)
            {
         printf ("Not enough memory available");
         exit ();
            }
     else
     {
          iniubuf (buffer, SIZE);
/*   Tells Unify where the buffer is and its size.*/

     if ((bseqacc(pack, first)) != 0)
```

```
/*   makes the first pack record current  */
     {
         printf ("there are no packs in the database\n");
         exit();
     }

     prmp(1,4,"number   charter_org              charter_date");
     prmp(50,4,"  council");
     prmp(1,5,"_____________________________________________");
     prmp(50,5,"___________");
     line_number=7;
     do
       {
       /* make the number of the pack current, using bfaccess to get
the pointer to the number from the buffer */

            bfaccess(pack,pack_num);
            pritm (line_number++);

     }
     while ((bseqacc (pack,next)) == 0);
     }
     return (&current_query);
}

pritm(x)

     /* a function used to specify the format of the output */

int x;

{
     pdata (3,x,pack_num);
     pdata (10,x,org);
     pdata (36,x,cdate);
     pdata (50,x,council);
}


main ()

{
/* the function call below is used to register the procedure for remote calls*/

rpc_initialize();

/* The code below was used to test the function on the host
computer, but is not used by the remote client at all, since
RPC only calls the numbered procedure. */

     if (current_query () != 0){
     printf ("\n");
     }
```

```
/* set up the server process */

svc_run();

fprintf(stderr, "Error:  svc_run should never return\n");
}
```

CHAPTER V
SYSTEM EVALUATION AND RESULTS

The expert system correctly performs the task of determining the required SQL query and displaying it on the screen. The designated query is then executed remotely using the Sun Microsystems Remote Procedure Call (RPC). The results of the query are displayed on the screen using the Lisp view-file function. Research continues into the proper procedure for executing the queries remotely. The expert knowledge part of the system works correctly to the degree that it was implemented. The system could easily be modified to formulate other queries through the addition of the appropriate rules. The system will not accept invalid responses for the yes-or-no questions or the type-query and type-info facts, however, it currently does not perform any range-checking of the pack or den numbers when they are entered. The user is requested to enter a number between 0 and 999 for the pack and between 0 and 9 for the den. Although the expert system does not check for numbers out of range, the Unify database does according to the types in Appendix A. Therefore, the database will return an error message or a message stating that no records were found. The expert system actually formulates ten different types of query for the user as depicted in the flowchart in

Figure 3-1, above. Since many of the queries depend on the user's input for values, the actual number of queries possible is much greater. Although the number of queries possible is virtually unlimited, it is normally the case that several types of query will be executed on a regular basis by a given user or group of users. If the expert system designer is provided with sufficient information, the appropriate query types could be implemented by the proposed expert system. It should be noted that different departments could be given different expert systems or the same expert system could be designed to ask what department the user was from in order to determine which set of queries to make available.

Sample Expert System Session. The following is a dribble file generated during a session with the scout-data expert system in an ART window on the AAIL.

```
=> reset

=> run


Hello.  My name is ART.  I can help you access the
         Cub Scout Data Base.

Do you wish to query the Data Base? (Y or N) No.
Goodbye, have a nice day.
 => run


Hello.  My name is ART.  I can help you access the
         Cub Scout Data Base.

Do you wish to query the Data Base? (Y or N) Yes.

Do You want information about a pack [P] or a den [D]?P
Do you want administrative information about the pack [A]
```

```
or information about the pack staff [S]? A
Do you want the information on all packs? (Y or N) Yes.
The required SQL query is

select *
from pack
order by number/

Here is the information you requested:

number|charter_org           |charter_date|council
-------------------------------------------------------
    83|Westside Christian Ch |    01/02/84|North Florida
   566|Diamond Elem. Sch     |    12/03/82|Savannah

Do you wish to query the Data Base again? (Y or N) Yes.

Do You want information about a pack [P] or a den [D]?D

Do you want administrative information about the den [A] or information about
the den members [M]? M

Do you want the information on all dens? (Y or N) No.

Please enter the Pack number [0-999]: 83

Do you want the information on all dens in the pack? (Y or N) No.

Please enter the Den number [0-9]: 3

The required SQL query is

select den_number,pack_number,position,first_name,MI
last_name,number,work_phone

from member m,people p, family f
where m.person_number=p.person_number and
m.family_number=p.family_number and
m.family_number=f.family_number and
pack_number=83 and
den_number=3
order by position,last_name/

Here is the information you requested:
           den_number|pack_number|position|first_name |MI|last_name  |number  |work_phone
           ------------------------------------------------------------------------------
                   3|         83|leadr   |Linda      |  |Davis      |378-6199|
                   3|         83|parnt   |John       |  |Davis      |378-6199|497-3045
                   3|         83|parnt   |Linda      |  |Davis      |378-6199|
                   3|         83|parnt   |Agustin    |  |Ortiz      |371-1930|335-8447
                   3|         83|parnt   |Kathy      |E |Ortiz      |371-1930|
                   3|         83|scout   |Jason      |  |Davis      |378-6199|
                   3|         83|scout   |Carlos     |R |Ortiz      |371-1930|
```

```
Do you wish to query the Data Base again? (Y or N) No.
Goodbye, have a nice day.
 => exit
```

The session above demonstrates the fact that even for a relatively complex query involving the joining of three tables, the expert system only needs to ask the user six questions in order to determine the required query. It was necessary to capture the session results in a dribble file because it is not currently possible to capture screen images on the printer and ART cannot be accessed from a remote terminal. It is, however, possible to capture the results of the session in a dribble file in order to demonstrate the firing of rules, as well as the assertion and retraction of facts. Another dribble file showing the rules and facts that were asserted and retracted for the above session is included as Appendix D.

Evaluation. The scout-data expert system produces ten types of SQL queries correctly after asking the user a few questions to determine the type of information that is desired. This eliminates the need for the user to be familiar with DBMS or the schema of the database being used. Even without the communications interface, the system serves a useful purpose in this regard. The full implementation of the expert system with a communication protocol and interface could potentially save an organization many hours of training together with the monetary costs associated with the training. Additionally, the queries could be executed faster because instead of the

user having to type them in, the expert system would be calling them as soon as the knowledge base required it. Using ART, it is very easy to modify the expert system to ask other questions or perform other actions. This makes it a very flexible system, thus increasing its potential usefulness.

CHAPTER VI
CONCLUSION

The use of an expert system on an artificial intelligence machine as a front end processor to interface with a relational database could potentially save an organization a lot of training time and money by eliminating the need for Database Management Systems (DBMS) training for those employees who will only need to use the database occasionally. The organization will still need a trained Database Administrator to maintain the DBMS, however, the expert system could include rules for allowing the input and updating of data as well as queries. The expert system knowledge base and rules can be custom-designed to the organization's requirement and as long as there is a physical communications link to the host computer on which the database resides and the appropriate software protocol is in place on both machines, the system can be fully implemented as described above. As previously stated, the expert system can formulate a complex SQL query after asking the user only a few simple questions. As long as the questions are designed in natural language that the potential users can understand, this is an effective way to operate the database, since it will allow untrained users to execute queries that formerly required DBMS training.

Since the rule-based actions are limited only by the amount of expertise and information provided to the knowledge engineer, this type of expert system could easily be used in almost any other area where a predefined set of actions are to be executed based upon the existence of some predefined conditions. The execution of the actual query would be faster using the expert system because the system would activate the query at the speed of the computer processor as soon as the knowledge base supported that action. Using an expert system would also reduce the number of errors caused by executing incorrect queries. This would result in additional savings of time and money.

Remote procedure calls had never been attempted previously on either of the computers involved in this research. This made it difficult to obtain local assistance in solving networking problems. As remote procedure calls become more common, the difficulties encountered in implementing the communications interface aspect of this project will greatly diminish as more computers will already have communications software installed and in use.

Future research in this area should include designing the expert system to completely generate the query by establishing facts and inheritances to determine exactly which attributes are desired, rather than merely executing "canned queries" that have been coded previously. Although the use of these queries is a limitation, it would not

normally be a major one even in the industry, because, usually, the same queries will be executed again and again. Also, the insertion, deletion and correction of data should be incorporated into future revisions of the system.

APPENDIX A
RELATIONAL DATABASE SCHEMA

Description

A database containing data about one or more Cub Scout Packs as well as the Dens and people (Staff and Den-members) associated with each pack was developed using the Unify database management system on the Computer and Information Sciences department's Gould powernode. There are several types of Staff-members: cubmaster, secretary, treasurer, committee chief and committee members. An adult may be a staff (committee) member as well as a Den member. There are three types of Den-members: scouts, leaders, and parents. An adult may be a leader as well as a parent and may be associated with more than one Den. Since there may be more than one member of a family associated with the pack, there is a need to store information about each individual separately. Using the Schema described below, it is possible to avoid duplication of the information common to all members of the same family in the database. By joining the Family and People tables, it is possible to obtain information pertaining to one or more members of any family. Although Unify supports the creation of data entry and query screens which allow a user to obtain information from multiple tables on a single screen, it also allows

users to call Structured Query Language (SQL) queries directly from the operating system (UNIX) shell as well as add and extract data from the database using C language programs[Un85a, Un85b, Un85c].

Relational Database Form (Schema)

Tables	Attributes (Key Fields Underlined)
Pack	NUMBER, CHARTER_ORG, CHARTER_DATE, COUNCIL
Den	DEN_NUMBER, PACK_NUMBER, MEETING_NIGHT, MEETING_LOCATION
People	PERSON_NUMBER, FAMILY_NUMBER, FIRST_NAME, MI, WORK_PHONE
Family	FAMILY_NUMBER, LAST_NAME, STREET_ADDR, CITY, STATE, ZIP_CODE, AREA_CODE, NUMBER
Staff	PERSON_NUMBER, FAMILY_NUMBER, PACK_NUMBER, POSITION
Members	PERSON_NUMBER, FAMILY_NUMBER, DEN_NUMBER, PACK_NUMBER, POSITION

Types	Attributes
DATE	CHARTER_DATE
NUMERIC (1)	DEN_NUMBER
NUMERIC (3)	PACK_NUMBER, FAMILY_NUMBER, AREA_CODE
NUMERIC (5)	ZIP_CODE
NUMERIC (9)	PERSON_NUMBER
STRING (1)	MI
STRING (2)	STATE
STRING (3)	MEETING_NIGHT
STRING (5)	POSITION
STRING (8)	NUMBER, WORK_PHONE
STRING (10)	CITY
STRING (11)	LAST_NAME, FIRST_NAME, MEETING_LOCATION
STRING (15)	COUNCIL, STREET_ADDR
STRING (22)	CHARTER_ORG

Entity-Relationship Model

Entity Sets

Pack

Den

Family

People

Weak Entity Types

Den depends on Pack

People depend on Family

Relationships. Below is a list of the relationships with the key attributes of each relation that participates in the relationship. The data types of the attributes are described below.

PACK-HAS-DEN	pnum
PEOPLE-BELONG-FAMILY	fid
PEOPLE-STAFF-PACK	spack, spid
DEN-MEMBERS-PEOPLE	unid, nam

Detailed List of Entity Sets and Attributes

NAME	ATTRIBUTE NAME	TYPE	SIZE	CONSTRAINTS
Pack				
	PACK_NUM	INTEGER	3	1-999, [NOT NULL]
	A unique 3 digit number identifier.			
	ORG	STRING	22	[NOT NULL]
	CDATE	DATE	NA	DD/MM/YY
Den				
	DEN_ID	COMBINED		
	NUMBER	INTEGER	1	1-9, [NOT NULL]
	PNUM	INTEGER	3	[NOT NULL] IN PACK.PACK_NUM
	NIGHT	STRING	3	IN ['MON', 'TUE', 'WED', 'THU', 'FRI', 'SAT', 'SUN']
	SITE	STRING	11	

People

PID	COMBINED		
PNO	INTEGER	9	0-999999999 [NOT NULL]
FID	INTEGER	3	1-999 [NOT NULL] IN FAMILY.FAM_ID
NAME	COMBINED		
FNAME	STRING	11	[NOT NULL]
MI	CHAR	1	
PHONE_NO	STRING	8	IN ['0'..'9','-']

Family

FAM_ID	INTEGER Numbers are unique within the DB.	3	[NOT NULL], 1-999
LNAME	STRING	11	[NOT NULL]
ADDRESS			
STREET	STRING	15	[NOT NULL]
CITY	STRING	10	[NOT NULL]
STATE	STRING	2	[NOT NULL]
ZIP	INTEGER	5	[NOT NULL]
PHONE			
AREA	INTEGER	3	[NOT NULL]
NUM	STRING	8	IN ['0'..'9','-']

Members

MEMID	COMBINED		
NAM	COMBINED		IN PEOPLE.PID [NOT NULL]
UNID	COMBINED		IN DEN.DEN_ID [NOT NULL]
MTYPE	STRING	5	[NOT NULL] IN ['SCOUT', 'PARNT', 'LEADR']

Staff

SID	COMBINED		
SPID	COMBINED		IN PEOPLE.PID [NOT NULL]
SPACK	COMBINED		IN PACK.PACK_NUM [NOT NULL]
STYPE	STRING	5	[NOT NULL] IN ['MASTR', 'COMMC', 'COMMM', 'SECRY', 'TREAS']

APPENDIX B
EXPERT SYSTEM SOURCE CODE

Listed below is the commented source file for the Cub Scout Data expert system. This file exists on the Applied Artificial Intelligence Laboratory (AAIL) Texas Instruments Explorer as AAIL: GUS; SCOUT-DATA.ART#>.

```
;;; -*- Mode: ART; Package: art-user; Base: 10.; Syntax:
Common-
;;;   lisp -*-
;;;
;;;
;;;   Cub Scout Data Base
;;;
;;;    Agustin Ortiz
;;;    Major, US Army Signal Corps
;;;
;;;    10 JUN 88
;;;
;;;
;;;    MASTER'S THESIS PROJECT
;;;
;;;    AN EXPERT SYSTEM TO ACCESS A RELATIONAL DATABASE
;;;
;;;

;;;   This program was written using the ZMACS editor and
;;;   compiled using ART V 3.0 on the Texas Instruments
;;;   Explorer.  The program asks the user several
;;;   questions in order to determine what type of query is
;;;   needed using structured query language (SQL).  The
;;;   command window is then cleared and the appropriate
;;;   SQL query is displayed.  When fully implemented, the
;;;   query will be executed against the Cub Scout database
;;;   located in the /cisg/grad/gus/scouts subdirectory of
;;;   the CIS Gould (host beach.cis.ufl.edu).
;;;
;;;
;;; Relations must first be defined before being used in an
;;; ART program.  This tells ART how many parameters to
;;; expect in order for error-checking to be performed
;;; automatically
```

```
(defrelation use-db (?boolean))

(defrelation type-query (?which))

(defrelation type-info (?which))

(deffacts initial-flag    ;Initial facts used to start the
system
  (use-db unknown)
  (type-query  unknown)
  (type-info unknown))

(defrule getting-started
   ?x<-(use-db ~Y & ~N)
   ;WHETHER to query the DB or NOT

    ;unclear or unknown.
         ;(Should only occur at the beginning of the
session.)
=>
   (set-interactive-mode nil)
;turn off warning messages
   (retract ?x)
                  ;remove unsatisfactory fact.
   (select-window 'command-window)
   (reshape-window #L'command-window 10 10 660 300)
   (clear-window #L'command-window)
   (printout t t t                          ;output to
screen.
        "Hello.  My name is ART.  I can help you access the
          Cub Scout Data Base." t t)
   (if (y-or-n-p (format nil "Do you wish to query the Data
Base? "))
       then
;assert a new fact according to the user response
       (assert (use-db Y))
       else
       (assert (use-db N))))
;;;
;;;
;;;  The following schemata can be used to show how
;;;  attributes are inherited.  Future plans are to use
;;;  them to make ART form its own queries using
;;;  inheritance.
;;;
;;;

(defschema subclass-of
       (instance-of inh-relation)
           (new-relations (is-a (?domain)(subclass)))
       (inverse has-subclasses))

(defschema has-subclasses
       (instance-of relation))
```

```
(defschema pack
       "Cub Scout Packs"       ;info about the pack
  (pack-number)
  (org)
  (charter-date)
  (council))

(defschema den
       "Cub Scout Dens"         ;info about the den

  (subclass-of pack)
  (den-number)
  (night)
  (site))

(defschema people
       "Individual People"      ;info about the individual
 (subclass-of family)
 (person-number)

 (first-name)
 (mi)
 (work-phone))

(defschema family
       "Families"
;info common to all members of the same family

 (family-number)
 (last-name)
 (street)
 (city)
 (state)
 (zip)
 (area)
 (home-phone))



(defrule query-db
       "An SQL query"
;information needed to query the data base

  (use-db Y)
  ?x<-(type-query ~P&~D)
=>
(retract ?x)
(printout t t t "Do You want information about a pack [P]
or a den [D]?")
(assert (type-query =(read))))

;;;
(defrule pack-query
```

```
     "determine whether info is desired about the pack,
itself, (admin) or about the people that run it (staff)"

  (use-db y)
  (type-query P)
  ?x<-(type-info ~A&~S)
=>
(retract ?x)
(printout t t "Do you want administrative information about
the pack [A] or information about the pack staff [S]? ")
(assert (type-info =(read))))


(defrule den-query
     "determine whether info is desired about the den,
itself, (admin) or about the people associated with it
(members)"

  (use-db y)
  (type-query D)
  ?x<-(type-info ~A&~M)
=>
(retract ?x)
(printout t t "Do you want administrative information about
the den [A] or information about the den members [M]? ")
(assert (type-info =(read))))

(defrule pack-admin
     "User wants admin info about the pack"
(use-db Y)
(type-query P)
(type-info A)
=>
(reshape-window #L'command-window 10 10 660 300)
(if (y-or-n-p (format nil "Do you want the information on
all packs? "))
    then
    (clear-window #L'command-window)
    (printout t t "The required SQL query is"
          t t "select *"
          t t "from pack"
          t t "order by number/" t t)
    (printout t t "Here is the information you requested:"
t t)
    (reshape-window #L'command-window 10 10 660 450)
;*********************
;This is the RPC call
    (callrpc "beach" 20118651 1 1 :xdr_void nil :xdr_string
"lm:gus;packs.text" :udp)
;*********************

    (view-file "lm:gus;packs.text")
    else
    (printout t t "Please enter the Pack number [0-999]: ")
```

```
     (bind ?number (read))
     (clear-window #L'command-window)
     (printout t t "The required SQL query is"
          t t "select *"
          t t "from pack"
          t t "where number=" ?number"/" t t)
     (if (= ?number 83)
     then
     (printout t t "Here is the information you requested:"
t t)
     (reshape-window #L'command-window 10 10 660 450)
;*********************
     (view-file "lm:gus;pack83-admin.text")
;This is where the RPC call belongs
     else                         (if (= ?number 566)
     then
          (printout t t "Here is the information you
requested:" t t)
          (reshape-window #L'command-window 10 10 660 450)
;*********************
          (view-file "lm:gus;pack-566-admin.text")
;This is where the RPC call belongs
          else
          (printout t t "But that pack is not in the
database.")))))

(defrule pack-staff
     "User wants staff info about the pack"
(use-db Y)
(type-query P)
(type-info S)
=>
(reshape-window #L'command-window 10 10 660 300)
(if (y-or-n-p (format nil "Do you want the information on
all packs? "))
    then
    (clear-window #L'command-window)
    (printout t t "The required SQL query is"
          t t "select pack_number,position,first_name,
MI,last_name,number,work_phone"
          t t "from staff s,people p, family f"
          t t "where s.person_number=p.person_number and"
          t t "s.family_number=p.family_number and"
          t t "s.family_number=f.family_number"
          t t "order by pack_number/" t t)
    (printout t t "Here is the information you requested:")
    (reshape-window #L'command-window 10 10 860 650)
;*********************
    (view-file "lm:gus;packs-staff.text")
;This is where the RPC call belongs

    else
    (printout t t "Please enter the Pack number [0-999]: ")
     (bind ?pnum (read))
```

```
     (clear-window #L'command-window)
     (printout t t "The required SQL query is"
          t t "select pack_number,position,first_name,
MI,last_name,number,work_phone"
          t t "from staff s,people p, family f"
          t t "where s.person_number=p.person_number and"
          t t "s.family_number=p.family_number and"
          t t "s.family_number=f.family_number and"
          t t "pack_number=" ?pnum"/" t t)
     (if (= ?pnum 83)
     then
     (printout t t "Here is the information you
requested:")
     (reshape-window #L'command-window 10 10 660 450)
;*********************
     (view-file "lm:gus;pack83-staff.text")
;This is where the RPC call belongs
     else
      (if (= ?pnum 566)
          then
          (printout t t "Here is the information you
requested:")
          (reshape-window #L'command-window 10 10 660 450)
;*********************
          (view-file "lm:gus;pack-566-staff.text")
;This is where the RPC call belongs
          else
          (printout t t "But that pack is not in the
database.")))))
(defrule den-admin
     "User wants admin info about the den"
(use-db Y)
(type-query D)
(type-info A)
=>
(reshape-window #L'command-window 10 10 660 300)
(if (y-or-n-p (format nil "Do you want the information on
all dens in the DB? ")
)
    then
    (clear-window #L'command-window)
    (printout t t "The required SQL query is"
          t t "select *"
          t t "from den"
          t t "order by pack_number,den_number/" t t)
    (printout t t "Here is the information you requested:"
t t)
    (reshape-window #L'command-window 10 10 660 450)
;*********************
    (view-file "lm:gus;dens-admin.text")

;This is where the RPC call belongs

    else
```

```
    (printout t t "Please enter the Pack number [0-999]: ")
     (bind ?pnum (read))
     (if (y-or-n-p (format nil "Do you want the information
on all dens in the pack? "))
     then
     (clear-window #L'command-window)
     (printout t t "The required SQL query is"
            t t "select *"
            t t "from den"
            t t "where pack_number=" ?pnum
            t t "order by den_number/" t t)
     (if (= ?pnum 83)
         then
         (printout t t "Here is the information you
requested:" t t)
         (reshape-window #L'command-window 10 10 660 450)
;*********************
         (view-file "lm:gus;dens-83-admin.text")
;This is where the RPC call belongs
         else
          (if (= ?pnum 566)
           then
           (printout t t "Here is the information you
requested:" t t)
           (reshape-window #L'command-window 10 10 660 450)
;*********************
           (view-file "lm:gus;dens-566-admin.text")
;This is where the RPC call belongs
           else
           (printout t t "But that pack is not in the
database.")))

     else
     (printout t t "Please enter the Den number [0-9]: ")
     (bind ?dnum (read))
     (clear-window #L'command-window)
     (printout t t "The required SQL query is"
            t t "select *"
            t t "from den"
            t t "where pack_number=" ?pnum " and"
            t t "den_number=" ?dnum"/" t t)
     (if (and (= ?pnum 83)
           (= ?dnum 3))
         then
         (printout t t "Here is the information you
requested:" t t)
         (reshape-window #L'command-window 10 10 660 450)
;*********************
         (view-file "lm:gus;den3-83-admin.text")
;This is where the RPC call belongs
         else
          (if (and (= ?pnum 83)
                (= ?dnum 9))
           then
```

```
          (printout t t "Here is the information you
requested:" t t)
          (reshape-window #L'command-window 10 10 660 450)
;*********************
          (view-file "lm:gus;den9-83-admin.text")
;This is where the RPC call belongs

          else
          (printout t t "But that den is not in the
database.")))))

(defrule den-member
     "User wants member info about the den"
(use-db Y)
(type-query D)
(type-info M)
=>
(reshape-window #L'command-window 10 16 780 450)
(if (y-or-n-p (format nil "Do you want the information on
all dens? "))
    then

    (clear-window #L'command-window)
    (printout t t "The required SQL query is"
          t t "select den_number,pack_number,position,
first_name,MI,last_name,number,work_phone"
          t t "from member m,people p, family f"
          t t "where m.person_number=p.person_number and"
          t t "m.family_number=p.family_number and"
          t t "m.family_number=f.family_number"
          t t "order by pack_number,den_number,position,
last_name/" t t)
    (printout t t "Here is the information you requested:"
t t)
    (reshape-window #L'command-window 10 10 800 750)
;*********************
    (view-file "lm:gus;dens-mem.text")
;This is where the RPC call belongs

    else
    (printout t t "Please enter the Pack number [0-999]: ")
     (bind ?pnum (read))
     (if (y-or-n-p (format nil "Do you want the information
on all dens in the pack? "))
     then

     (clear-window #L'command-window)
     (printout t t "The required SQL query is"
           t t "select den_number,pack_number,position,
first_name,MI,last_name,number,work_phone"
           t t "from member m,people p, family f"
           t t "where m.person_number=p.person_number and"
           t t "m.family_number=p.family_number and"
           t t "m.family_number=f.family_number and"
```

```
            t t "pack_number=" ?pnum
            t t "order by den_number,position,last_name/" t
t)
     (if (= ?pnum 83)
         then
         (printout t "Here is the information you
requested:")
         (reshape-window #L'command-window 10 10 800 700)
;*********************
         (view-file "lm:gus;dens83-mem.text")
;This is where the RPC call belongs
         else
          (if (= ?pnum 566)
           then
           (printout t t "Here is the information you
requested:")
           (reshape-window #L'command-window 10 10 850 550)
;*********************
           (view-file "lm:gus;dens-566-mem.text")
;This is where the RPC call belongs
           else
           (printout t t "But that pack is not in the
database.")))
     else
     (printout t t "Please enter the Den number [0-9]: ")
     (bind ?dnum (read))

     (clear-window #L'command-window)
     (reshape-window #L'command-window 10 10 850 560)
     (printout t t "The required SQL query is"
            t t "select den_number,pack_number,position,
first_name,MI,last_name,number,work_phone"
            t t "from member m,people p, family f"
            t t "where m.person_number=p.person_number and"
            t t "m.family_number=p.family_number and"
            t t "m.family_number=f.family_number and"
            t t "pack_number=" ?pnum " and"
            t t "den_number=" ?dnum
            t t "order by position,last_name/" t t)
     (if (and (= ?pnum 83)
           (= ?dnum 3))
         then
         (printout t t "Here is the information you
requested:")
         (reshape-window #L'command-window 10 10 850 600)
;*********************
         (view-file "lm:gus;den3-83-mem.text")
;This is where the RPC call belongs
         else
          (if (and (= ?pnum 83)
                (= ?dnum 9))
           then
           (printout t t "Here is the information you
requested:")
```

```
          (reshape-window #L'command-window 10 10 850 660)
;*********************
          (view-file "lm:gus;den9-83-mem.text")
;This is where the RPC call belongs

          else
          (printout t t "But that den is not in the
database.")))))

(defrule another-query
     "Find out if user wants to make another query."

  (declare (salience -1))
;make this wait until other rules have fired
  ?use-db<-(use-db Y)
  ?type-query<-(type-query ?D & ~unknown)
  ?type-info<-(type-info ?M & ~unknown)
=>

(if (y-or-n-p (format nil "

Do you wish to query the Data Base again? "))
       then
       (retract ?use-db
         ?type-query
         ?type-info)
       (clear-window #L'command-window)
       (assert (use-db Y)
           (type-query  unknown)
           (type-info unknown))
       else
       (retract ?use-db)
       (assert (use-db N))))

(defrule lets-quit
     "User doesn't want to query the data base"
(use-db N)
=>
(reshape-window #L'command-window 19 16 398 166)
(clear-window #L'command-window)
(printout t t "Goodbye, have a nice day.")
(halt)
(reset))
```

APPENDIX C
USER'S MANUAL FOR EXPERT SYSTEM

The following procedure must be followed in order to use the Cub Scout database expert system on the Texas Instruments Explorer in the Applied Artificial Intelligence Laboratory (AAIL).

Cold-Boot and Login. The system should be cold-booted if the previous user failed to do so. This can be done by pressing META-CONTROL-META-CONTROL-RUBOUT. This is not necessary if other processes have been started previously by the same user since the last time the system was cold-booted. If this is the first application to be run after cold-booting, the operating system will ask the user to login by typing "(login 'yourname)".

Accessing the Automated Reasoning Tool (ART). To access ART, regardless of what the user had been doing previously, the SYSTEM key should be pressed followed by the letter 'A'. The first time this is done, ART will be initialized and the root menu will be displayed in the upper right corner of the screen with the command window in the upper left corner of the screen.

Loading AAIL:GUS;SCOUT-DATA. The load command may be typed directly into the command window followed by a carriage return or it may be selected by clicking the left

button of the mouse while highlighting 'load'. Either way, the system will respond by requesting the filename to be loaded. The name above will load the newest version of SCOUT-DATA.ART, the file containing the definitions, rules and facts for the expert system.

Resetting the Knowledge Base. Prior to running the expert system, ART must be reset. Again, this command may be typed into the command window or selected with the left mouse button. When the cursor returns, the system is ready for use.

Running Scout-Data. The run command also may be entered using either the keyboard or the mouse. After the run command is issued, the command window will be cleared and reshaped. The following initial message will be displayed in the command window:

> Hello. My name is ART. I can help you access the Cub Scout data base.
>
> Do you wish to query the database? (Y or N)

The only valid responses are 'y' and 'n'. If the user responds with 'n', ART clears and reshapes the command window, displays a farewell message and resets the knowledge base. If the user responds with 'y', ART responds by asking if the user wants information about Packs or Dens. All other inputs besides 'p' and 'd' are rejected. According to the type of query desired, ART will ask whether administrative information or information about the people associated with the pack is desired. Once this

information has been supplied, again, by entering a single letter from the supplied choices, ART will ask whether the information is to be about all packs (dens) in the data base, all the dens in a specific pack (after asking for the pack number), or a specific den (by asking for the den number). Once all of the information has been supplied by the user, the command window is cleared, reshaped, if necessary, and the required Structured Query Language (SQL) query is displayed in the command window. The designated query should be executed against the database via the ETHERNET and the results displayed in the command window. ART will then ask the user if he or she wants to execute another query. If the user responds 'n' ART displays the farewell message and resets the knowledge base as above. For 'y' ART will clear the command window and ask the user what type query is desired.

APPENDIX D
DRIBBLE FILE OF SAMPLE SESSION

The file listed below is an extract of a dribble file obtained by running the sample ART session described in Chapter V while watching facts and rules. This option allows the user to see which rules are fired as well as which facts are asserted and retracted by each rule. In the file below, the following symbols are used:

==>	indicates a fact asserted
<==	indicates a retracted fact

```
 => load
File name: gus;scout-data

Loading AAIL: GUS; SCOUT-DATA.ART#74 in package ART-USER
and base 10.
Compiling rule GETTING-STARTED... +P+J
Booting the schema system...
Compiling schema SUBCLASS-OF...
Compiling schema HAS-SUBCLASSES...
Compiling schema PACK...
Compiling schema DEN...
Compiling schema PEOPLE...
Compiling schema FAMILY...
Compiling rule QUERY-DB... +P+P+J
Compiling rule PACK-QUERY... =P+P+J+P+J
Compiling rule DEN-QUERY... =P+P+J+P+J
Compiling rule LETS-QUIT... +P+J
Compiling rule PACK-ADMIN... =P=P=J+P+J
Compiling rule PACK-STAFF... =P=P=J+P+J
Compiling rule DEN-ADMIN... =P=P=J=P+J
Compiling rule DEN-MEMBER... =P=P=J+P+J
Compiling rule ANOTHER-QUERY... =P+P+J+P+J
 => reset

 => watch
```

```
 => rules

 => facts

 => pop

 => run


FIRE 0 GETTING-STARTED (f-1302,)
<==  f-1302 [USE-DB UNKNOWN]

Hello.  My name is ART.  I can help you access the
          Cub Scout Data Base.

Do you wish to query the Data Base? (Y or N) No.
==>  f-1306 [USE-DB N]

FIRE 1 LETS-QUIT (f-1306,)
Goodbye, have a nice day.


==>  f-1301 [|InitialFact|]
==>  f-1302 [USE-DB UNKNOWN]
==>  f-1303 [TYPE-QUERY UNKNOWN]
==>  f-1304 [TYPE-INFO UNKNOWN]
 => run


FIRE 1 GETTING-STARTED (f-1302,)
<==  f-1302 [USE-DB UNKNOWN]

Hello.  My name is ART.  I can help you access the
          Cub Scout Data Base.

Do you wish to query the Data Base? (Y or N) Yes.
==>  f-1305 [USE-DB Y]

FIRE 2 QUERY-DB (f-1305,f-1303)
<==  f-1303 [TYPE-QUERY UNKNOWN]

Do You want information about a pack [P] or a den [D]?P

==>  f-1306 [TYPE-QUERY P]

FIRE 3 PACK-QUERY (f-1305,f-1306,f-1304)
<==  f-1304 [TYPE-INFO UNKNOWN]

Do you want administrative information about the pack [A]
or information about the pack staff [S]? A

==>  f-1307 [TYPE-INFO A]

FIRE 4 PACK-ADMIN (f-1305,f-1306,f-1307)
```

```
Do you want the information on all packs? (Y or N) Yes.
The required SQL query is

select *

from pack

order by number/


Here is the information you requested:

number|charter_org            |charter_date|council
------------------------------------------------------
    83|Westside Christian Ch  |    01/02/84|North Florida
   566|Diamond Elem. Sch      |    12/03/82|Savannah


FIRE 5 ANOTHER-QUERY (f-1305,f-1306,f-1307)

Do you wish to query the Data Base again? (Y or N) Yes.

<==  f-1305 [USE-DB Y]
<==  f-1306 [TYPE-QUERY P]
<==  f-1307 [TYPE-INFO A]
==>  f-1308 [USE-DB Y]
==>  f-1309 [TYPE-QUERY UNKNOWN]
==>  f-1310 [TYPE-INFO UNKNOWN]

FIRE 6 QUERY-DB (f-1308,f-1309)
<==  f-1309 [TYPE-QUERY UNKNOWN]

Do You want information about a pack [P] or a den [D]?D

==>  f-1311 [TYPE-QUERY D]

FIRE 7 DEN-QUERY (f-1308,f-1311,f-1310)
<==  f-1310 [TYPE-INFO UNKNOWN]

Do you want administrative information about the den [A]
or information about the den members [M]? M

==>  f-1312 [TYPE-INFO M]

FIRE 8 DEN-MEMBER (f-1308,f-1311,f-1312)

Do you want the information on all dens? (Y or N) No.

Please enter the Pack number [0-999]: 83

Do you want the information on all dens in the pack? (Y or
N) No.
```

```
Please enter the Den number [0-9]: 3

The required SQL query is

select den_number,pack_number,position,first_name,
MI,last_name,number,work_phone

from member m,people p, family f

where m.person_number=p.person_number and

m.family_number=p.family_number and

m.family_number=f.family_number and

pack_number=83 and

den_number=3

order by position,last_name/


Here is the information you requested:

den_number|pack_number|position|first_name |MI|last_name |number  |work_phone
-------------------------------------------------------------------------
        3|         83|leadr   |Linda      |  |Davis     |378-6199|
        3|         83|parnt   |John       |  |Davis     |378-6199|497-3045
        3|         83|parnt   |Linda      |  |Davis     |378-6199|
        3|         83|parnt   |Agustin    |  |Ortiz     |371-1930|335-8447
        3|         83|parnt   |Kathy      |E |Ortiz     |371-1930|
        3|         83|scout   |Jason      |  |Davis     |378-6199|
        3|         83|scout   |Carlos     |R |Ortiz     |371-1930|


FIRE 9 ANOTHER-QUERY (f-1308,f-1311,f-1312)

Do you wish to query the Data Base again? (Y or N) No.
<==  f-1308 [USE-DB Y]
==>  f-1313 [USE-DB N]

FIRE 10 LETS-QUIT (f-1313,)
Goodbye, have a nice day.

          .
          .
          .

<==  f-1311 [TYPE-QUERY D]
<==  f-1312 [TYPE-INFO M]
<==  f-1313 [USE-DB N]

          .
```

```
          .
          .

==>  f-1301 [|InitialFact|]
==>  f-1302 [USE-DB UNKNOWN]
==>  f-1303 [TYPE-QUERY UNKNOWN]
==>  f-1304 [TYPE-INFO UNKNOWN]
 => exit
```

APPENDIX E
NOTES ON NETWORKING AND REMOTE PROCEDURE CALLS

The use of two different computers required familiarization with the operating systems of both and how they support network operations. This appendix contains observations about the requirements and capabilities of both computers involved in this project. In order to transfer data effectively between two computers, a data path must be established, either directly or indirectly. Additionally, the following tasks must be performed:

1. The source system must either activate the direct data communication path or inform the communication network of the identity of the desired destination system.

2. The source system must ascertain that the destination system is prepared to receive data.

3. The file transfer application on the source system must ascertain that the file management program on the destination system is prepared to accept and store the file.

4. If the file formats used on the two systems are incompatible, one or the other system must perform a format translation function.

The actions described above are commonly referred to as computer communications. The set of computer stations interconnected by means of a communication network is referred to as a computer network[ST87a]. Five types of computer networking facilities have been established:

Private Links

Private meshed networks

Public switched networks

Public data networks and value-added networks

Local-area networks[Bo83]

Two concepts must be understood in order to discuss computer networks:

Protocols

Computer-communications architecture[St87a]

Protocols are rules that have been established to define how information is to be formatted for transmission as well as the command syntax for a particular network transaction[Bo83].

A computer-communications architecture is the set of protocols and processes needed in order for communications to be carried out. Figure E-1 below illustrates the layered hierarchy of structured protocols.

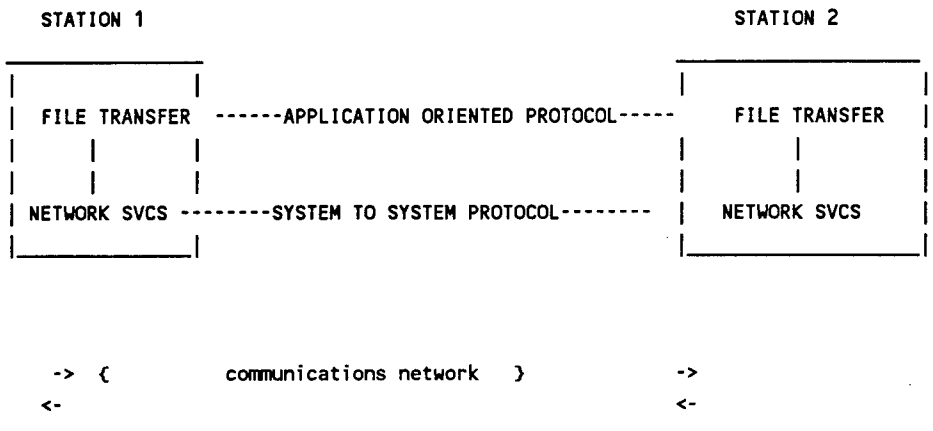

FIGURE E-1. Computer Communications Architecture [ST87a]

Gould Powernode

The Gould Powernode uses the Berkeley UNIX operating system. Although the earliest Unix implementations were

not good at interprocess communication, almost all current Unix systems support the UUCP (Unix to Unix Communications Protocol). 4.2BSD supports the DARPA Internet protocols UDP, TCP, IP, ICMP on many Ethernet, token ring and Arpanet interfaces. The operating system kernel makes communications protocols available through the socket system call which was originally written by Gurwitz of BBN[Pe85].

In the Open Systems Interconnection (OSI) model developed in 1984 by the international Organization for Standardization (ISO), a protocol implementation may only communicate with another entity that uses the same protocol at the same layer, or with the interface to the next protocol layer above or below in the same system[St87a]. This model was designed to set the standard for new communications networks and is currently being used on many applications by the U. S. government.

The Unix networking support is more closely related to the Arpanet Reference Model (ARM). ARM is a predecessor of the OSI model. The ARM consists of the following three protocol layers:

Process/Applications

Host-Host

Network interface

A user's process must communicate with network protocols using socket to set up and control communications. This corresponds to the session layer of OSI. The process may access any layer of protocol if it uses the raw socket

type. The actual transfer of data is usually done using buffers called mbufs. Figure E-2 below shows a comparison of the network reference model layers.

OSI MODEL	ARPANET MODEL	4.2BSD LAYERS	EXAMPLE LAYERING
Application		User Programs	telnet
Presentation	Process/	and Libraries	
Session	Applications	Sockets	SOCK-STREAM
			TCP
Transport	Host-Host	Protocols	IP
Network			
	Network	Network	Ethernet
Data Link	Interface	Interface	Driver
Hardware	Network	Network	Interlan
	Hardware	Hardware	Controller

Figure E-2. Network Reference Models and Layering [Pe85]

The Department of Defense (DOD) issued a set of standard protocols as listed in Figure E-3, below:

MIL-STD-1777 Internet Protocol (IP)
Provides a connectionless service for end systems to communicate across one or more networks. Does not assume the networks to be reliable.

MIL-STD-1778 Transmission Control Protocol (TCP)
A reliable end-to end data transfer service. Equivalent to the ISO Class 4 transport protocol.

MIL-STD-1780 File Transfer Control Protocol (FTP)
A simple application for transfer of ASCII, EBCDIC, and binary files.

MIL-STD-1781 Simple Mail Transfer Protocol (SMTP)
A simple electronic mail facility.

MIL-STD-1782 TELNET Protocol
Provides a simple scroll-mode terminal capability.

Figure E-3. DOD Military Standard Protocols

There are two major applications that can be carried out

between stations on the Arpanet: File Transfer Protocol(FTP) and TELNET. FTP provides a way to transfer files between two computers while TELNET enables a user to perform a remote login to a distant computer. A user obtains the FTP function via an unprivileged process. This FTP program is only called when there is traffic to be sent, terminating upon completion of the transfer. The FTP process opens a TCP connection to the desired destination and creates another data process to assist with the management of the transfer. When the transaction is completed, the server closes the connection, signalling the waiting user. The Unix server has a process that is created when the system is booted and stays in a sleeping state until signalled by an inbound request for FTP. When this happens, the server creates a new process to take care of the transaction, then goes back to sleep[St88].

The process described above is largely transparent to the user. In order to invoke the desired function, the user merely enters "ftp hostname" or "telnet hostname".

Texas Instruments Explorer

The Explorer supports network operations in several protocols including CHAOSNET, INTERNET PROTOCOL (IP), TRANSMISSION CONTROL PROTOCOL (TCP), TELNET, FILE TRANSFER PROTOCOL (FTP). The network protocols are installed as FLAVORS intermixed to achieve the effect of the seven layers of the Open Systems Interconnection (OSI) standard.

Because of the way FLAVORS are implemented, it is possible for a user to access the network services in many different ways, either interactively or from within a program. The multiprocessing capability of the Explorer allows more than one network process to be executed simultaneously. VT100 emulation is available as a separate process, although it does, in fact, run as a process above the TELNET protocol, which likewise is running TCP/IP[Ti87]. The Explorer also supports the RPC procedure, although in order to use it, the user must issue a call to "make-system" after logging in to the system: (make-system 'rpc :noconfirm :silent).

Remote Procedure Call (RPC)

RPC was developed by Sun Microsystems to allow different computers using different software to interact at the procedure level on any network. Using RPC, a computer can call a procedure on another computer, passing one argument to the procedure, and receive the value returned by the procedure. The External Data Representation (XDR) protocol is used to ensure the compatibility of data between machines. The two parts of the RPC protocol are listed below:

Caller	calls a procedure on a remote host
Server	services RPC requests on the remote host

External Data Representation (XDR). The XDR protocol developed by Sun Microsystems permits computers to exchange operands over a network even though they may use different

word lengths, floating point representations or byte orders. In order to make this possible, the sending machine must filter its data types into a standard representation to be output to the network. The receiving machine, in turn, filters the data received from the network into its native data types. Both machines must use the same XDR types in the same order for a transfer to be successful. The filters for primitive data types are included as part of RPC, but the user can also build custom filters, if desired.

Client. In order to make a remote procedure call, the following steps must be followed:

1. Find out the program number, procedure number and version number of the procedure to be called.

2. Determine the appropriate filter type for the argument to be passed to the procedure.

3. Call the procedure with the required argument, then check for the occurrence of errors.

4. Use the value returned in the local program.

Server. A programmer at the remote host must perform the following functions:

1. Select the correct filter type for the argument of the procedure call

2. Write the procedure. It must accept only one argument and return only one value.

3. Register the procedure for being called remotely.

Filter Types. Listed below are some of the filter types available for use with XDR:

LISP	XDR	C
:XDR-INTEGER	INTEGER	XDR-INT

		XDR-LONG
		XDR-SHORT
:XDR-FLOAT	FLOAT	XDR-FLOAT
:XDR-STRING	STRING	XDR-STRING

Registerrpc. The registerrpc function is used to register a procedure so that it may be called from a remote machine. It is generally invoked as part of the main C program in which the procedure is defined as illustrated below:

```
registerrpc(prog#, vers#, proc#, function-name, xdr-in, xdr-out)
```

Callrpc. The callrpc function is used to make the actual call to the remote procedure using the syntax below:

```
callrpc host prog# vers# proc# xdr-in in xdr-out out &optional credentials (protocol :udp)
```

The arguments of the function call are explained below:

host: identifies the remote computer (e.g. BEACH)

prog#: a 32 bit unsigned integer to identify the remote program
vers#: a 32 bit unsigned integer to identify the version number, if applicable, of the program

proc#: a 32 bit unsigned integer to identify which procedure within the remote program to execute

xdr-in: the XDR filter function to be used for encoding into the net

in: a pointer to the object to be passed to the remote procedure

xdr-out: the XDR filter function to be used for decoding from the net

out: a pointer to the object into which the result of the remote procedure will be passed

protocol: :udp for User Datagram Protocol[Te88]

REFERENCES

Bo83 Grayce M. Booth, The Design of Complex Information Systems, McGraw-Hill, New York, 1983.

Bo86 Boy Scouts of America, Wolf Cub Scout Book, Boy Scouts of America, Irving, Texas, 1986.

Ch87 Tae Gyu Chang, "Development of an Expert System for Multichannel EEG Signal Analysis," Ph.D. Dissertation, University of Florida, 1987.

Cl87 Bruce D. Clayton, ART Programming Tutorial, 4 Vols., Inference Corporation, Los Angeles, California, 1987.

Da86 C. J. Date, An Introduction to Database Systems, Vol. 1, 4th ed., Addison-Wesley, Reading, Massachusetts, 1986

Em87 John Emrich, "Remote File Systems, Streams, and Transport Level Interface," In UNIX Papers for UNIX Developers and Power Users, ed. Mitchell Waite, Howard W. Sams & Company, Indianapolis, 1987, 260 - 305.

Ge83 W. B. Gevarter, "Expert Systems: Limited But Powerful," IEEE Spectrum, Vol. 20, No. 8, Aug. 1983, 39-44.

Gr87 John Grant, Logical Introduction to Databases, Harcourt Brace Jovanovich, Orlando, Florida, 1987.

Ha85 F. Hayes-Roth, "Rule-Based Expert Systems," Communications of the ACM, Vol. 28, No. 9, Sep. 1985, 921-932.

Ha83 F. Hayes-Roth, D. A. Waterman, and D. B. Lenat, Building Expert Systems, Addison-Wesley, Reading, Massachusetts, 1983.

Le85 C. N. Lee, "Expert System Design and Implementation for Multichannel Sleep EEG Signal Processing," Ph.D. Dissertation, University of Florida, 1985.

Mc88 Richard McCurdy, "UFNET the University of Florida Broadband Network", unpublished information paper distributed to students and faculty, University of Florida, 1988.

Mc82 J. McDermott, "R1: A Rule-based Configurer of Computer Systems," Artificial Intelligence, Vol. 19, 1982, 39-88.

Pe85 James L. Peterson and Abraham Silberschatz, Operating Systems Concepts, 2d Ed., Addison-Wesley, Reading, Massachusetts, 1985.

Pr85 D. S. Prerau, "Selection of an Appropriate Domain for an Expert System," The AI Magazine, Summer 1985, 26-30.

Ra87 Eric Raymond, "The Future of UNIX and Open System Standards," In UNIX Papers for UNIX Developers and Power Users, ed. Mitchell Waite, Howard W. Sams & Company, Indianapolis, 1987, 486 - 504.

Sp87 Charles Spurgeon, "Ethernet: A UNIX LAN," In UNIX Papers for UNIX Developers and Power Users, ed. Mitchell Waite, Howard W. Sams & Company, Indianapolis, 1987, 306 - 341.

St87a William Stallings, The Open Systems Interconnection (OSI) Model and OSI-Related Standards, Handbook of Computer Communications Standards, Macmillan, New York, 1987.

St87b William Stallings, Local Network Standards, Handbook of Computer Communications Standards, Macmillan, New York, 1987.

St88 William Stallings, Department of Defense (DOD) Protocol Standards, Handbook of Computer Communications Standards, Macmillan, New York, 1988.

St82 M. Stefik, J. Aikins, R. Balzer, J. Benoit, L. Birnbaum, F. Hayes-Roth, and E. Sacerdoti, "The Organization of Expert Systems, A Tutorial," Artificial Intelligence, Vol. 18, No. 2, 135-173.

Su86 Sun Microsystems, Incorporated, "RPC System Manual," Sun Microsystems, Mountain View, Calif., 1986.

Te87 Texas Instruments, "Explorer System Manuals", Texas Instruments, Incorporated, Austin, Texas, 1987.

Te88 Texas Instruments, "microExplorer Programming", Texas Instruments, Incorporated, Austin, Texas, 1988.

Un85a Unify Corporation, Unify Relational Data Base Management System Programmers' Manual, Unify Corporation, Lake Oswego, Oregon, 1985.

Un85b Unify Corporation, Unify Relational Data Base Management System Reference Manual, Unify Corporation, Lake Oswego, Oregon, 1985.

Un85c Unify Corporation, Unify Relational Data Base Management System Tutorial Manual, Unify Corporation, Lake Oswego, Oregon, 1985.

Wa86 D. A. Waterman, A Guide to Expert Systems, Addison-Wesley, Reading, Massachusetts, 1986.

Wi84 Patrick Henry Winston, Artificial Intelligence, Addison-Wesley, Reading, Massachusetts, 1984.

BIOGRAPHICAL SKETCH

Agustin Ortiz, Jr., was born on July 11, 1954, in San Juan, Puerto Rico. He graduated in June 1972 from Antilles High School, Fort Buchanan, Puerto Rico. He received his Bachelor of Science degree in June 1976 from the United States Military Academy, West Point, New York, and was commissioned in the Regular Army Signal Corps where he currently holds the rank of Major. He was certified as an Engineer in Training by the Commonwealth of Pennsylvania in September 1976. He is a graduate of the Jungle Warfare School, Basic Airborne Course, Signal Officer Basic Course, Jumpmaster Course, Special Forces Officer Course, Signal Officer Advanced Course, Recruiting Commanders Course, Unit Discussion Leaders Course, Combined Arms and Services Staff School, TACFIRE Command and Staff Course and the United States Army Command and General Staff Officer Course. He is fluent in Spanish as well as English and has conversational knowledge of German. His military awards include the Jungle Expert Badge, the National Defense Service Medal, the Master Parachute Badge, the Army Service Ribbon, the Army Overseas Ribbon, and three awards of the Army Commendation Medal. He is a member in good standing of the Association for Computing Machinery. He

has been attending the University of Florida in pursuit of the Master of Science degree in electrical engineering since August 1986. After graduation from the University of Florida, Major Ortiz will be assigned to the United States Army Computer Engineering Center, Fort Huachuca, Arizona.

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

A. Antonio Arroyo, Chairman
Associate Professor of Electrical Engineering

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

Donald G. Childers
Professor of Electrical Engineering

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

Jose C. Principe
Associate Professor of Electrical Engineering

This thesis was submitted to the Graduate Faculty of the College of Engineering and to the Graduate School and was accepted as partial fulfillment of the requirements for the degree of Master of Science.

August 1988

Dean, College of Engineering

Dean, Graduate School